NASA Technical Memorandum 86264

A ROUTE GENERATOR CONCEPT FOR AIRCRAFT ONBOARD FAULT MONITORING

MICHAEL T. PALMER
KATHY H. ABBOTT

AUGUST 1984

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

SUMMARY

Because of the increasingly complex environments in which the flight crews of commercial aviation aircraft must operate, a research effort is currently underway at NASA Langley Research Center to investigate the potential benefits of intelligent cockpit aids, and to establish guidelines for the application of artificial intelligence techniques to advanced flight management concepts. The segment of this research area that concentrates on automated fault monitoring and diagnosis requires that a reference frame exist, against which the current state of the aircraft may be compared to determine the existence of a fault. This paper describes a computer program which generates the portion of that reference frame that specifies the horizontal flight route.

INTRODUCTION

In the commercial aviation industry, safety and efficiency are of utmost importance. Flight crews must operate in increasingly complex environments, and automation in the form of intelligent cockpit aids may assist them in improving safety and efficiency. Modern microprocessor and display technology have made it feasible to automate many flight deck functions, and the advent of expert systems technology has proven that artificial intelligence (AI) has a great deal to offer to enhance and aid the man in the system. In particular, automation of fault monitoring and diagnosis is desirable to ensure safe and efficient operations, since response to faults on board an aircraft is often time critical. It is believed that the use of artificial intelligence concepts can facilitate this automation.

An effort is currently underway at NASA Langley Research Center to investigate the potential benefits of intelligent cockpit aids, and to establish guidelines for the application of AI techniques to advanced flight management concepts. One application area within this research effort is onboard fault monitoring and diagnosis. The segment of this area that concentrates on automated fault monitoring requires that the actual aircraft state be compared to some desired reference aircraft state in order to detect the existence of a fault.

The aircraft state has been divided into portions to facilitate the generation of the desired reference aircraft state. A computer program has been developed to generate the portion of this reference state that specifies a horizontal route minimizing ground distance between the origin and destination airports using the established airway system designated by the Federal Aviation Administration and in use by Air Traffic Control. This report documents this computer program and describes its use and capabilities.

BACKGROUND

The initial concepts for the fault monitoring and diagnosis segment of the intelligent flight management research were developed at the University of

Illinois at Urbana-Champaign under a grant from NASA Langley Research Center (ref. 1). The major thrust of the work was to explore the existing artificial intelligence techniques applicable to fault monitoring and diagnosis within the flight domain and, if necessary, develop new techniques more suited to this application. Out of this research came the basic concepts for a multi-level architecture within which the fault monitoring segment would operate. The description and precise function of each of the levels within this architecture have evolved as new ideas were incorporated.

The basic architecture of the fault monitoring segment is organized as a hierarchy. The hierarchy consists of four distinct but interconnected levels as shown in Figure 1: the Route Level, the Trajectory Level, the Flight Dynamics Level, and the Subsystems Level. In general, each level is subservient to the one above it, supplying the means of achieving the goals which are passed to it from above. This holds true except in the case of the Subsystems Level; the relationship here could be more accurately described as a precondition-satisfaction check. An expert system is included within this framework to handle all diagnostic duties. The system architecture, the monitoring functions, and the expert system are written in the LISP language dialect known as MACLISP, and the Route and Trajectory Level Generators are written in FORTRAN 77. The following is a general description of each level within the fault monitoring and diagnosis segment of the intelligent flight management research area.

The Route Level

The Route Level is the highest level in the architecture, passing along its goals and restrictions to the Trajectory Level. These primary flight goals include the generated route and any time, weather, traffic, or other Air Traffic Control (ATC) restrictions. The Route Level consists of two main sections, the Route Generator and the Route Monitor. The Route Generator generates the shortest route from the city of departure to the city of arrival, taking into consideration constraints such as weather conditions and ATC restrictions. The Route Monitor continuously monitors the progress of the flight to determine if the generated route is being followed.

The Trajectory Level

This level also consists of two sections, the Trajectory Generator and the Trajectory Monitor. The Trajectory Generator utilizes the primary flight goals sent down from the Route Level Generator. Once these are received, the Trajectory Generator produces an optimal trajectory that minimizes either fuel or direct operating costs. The generated trajectory is passed down to the Flight Dynamics Level. The Trajectory Monitor monitors the progress of the flight from the vertical- and time-based viewpoint, deciding whether or not the optimal trajectory is being properly followed.

The Flight Dynamics Level

This level uses an internal model of the aircraft to generate all necessary control inputs such as wheel and column positions, throttle positions, and flap settings that are required to achieve the desired trajectory. This sequence of control inputs is then transferred down to the Subsystems Level.

The Subsystems Level

The Subsystems Level checks the control input sequence and amplitude to assure that the inputs do not attempt to extend the aircraft beyond its normal operational and performance limits. This level contains the knowledge about the aircraft subsystems, and checks that each subsystem is working properly.

When the checks at the Subsystem Level have been completed, the Flight Dynamics Level passes its sequence of control inputs back up into the Trajectory Level, which then passes the generated optimum trajectory back up to the Route Level. Here it is stored for referencing by the route monitoring functions. Once the flight has been initiated, constant surveillance of all critical parameters is maintained, and if any condition is observed that has caused or will cause deviation from the planned route, the expert system is activated. The expert system itself currently uses a rule-based approach to problem solving as described in ref. 2, but this will later be expanded to more closely combine the functions of error detection and fault diagnosis into a more unified effort.

DESCRIPTION OF THE ROUTE GENERATOR

The Route Generator is designed to run prior to each flight so that all current information on weather, traffic, and Air Traffic Control (ATC) instructions can be used. If any of these initial conditions or instructions are changed during the flight so as to render the previously generated route unacceptable, the Route Generator can be executed again using the next waypoint of the path as the point of departure, and the destination airport, whether different or the same as originally planned, as the point of arrival. This inflight re-routing capability is the principal advantage of a program which actively generates the route rather than retrieving a rigid, preplanned course from a storage device. The Route Generator itself is comprised of two major sections, the Selector and the Compiler.

The Route Selector

The Route Selector is the section which chooses the shortest allowable path from origin to destination using the established High Altitude Jet Route System. For the purposes of analysis, the Jet Route System is represented within this program in graph form, the nodes representing geographical reference points, or waypoints, and the arcs between these nodes denoting the airway segments. Each arc must be assigned a weighting factor. For the case

of locating the shortest path between any two nodes in the graph, this weighting factor is represented by the mileage between waypoints. The matrix containing the distance information is arranged so that arcs are identified by their location within the two-dimensional graph. The index for each dimension is a pointer to the name of the corresponding node (waypoint) in the one-dimensional matrix containing the node identifiers. Thus the graph, when printed out, takes the form of a mileage chart, with only those mileages shown that represent existing segments. A sample portion of the graph is shown in figure 2. The subroutine which performs the actual selection of the sequence of arcs from origin to destination nodes uses Dijkstra's algorithm (ref. 3). This algorithm locates the path of least resistance through an array of weighting factors, and is ideally suited for application to this particular class of problem.

The Route Compiler

The Route Compiler assembles the sequence of track angles (no-wind true headings) and the corresponding distances required to follow the chosen path through the established system of jet airways, as well as the magnetic variation from true north associated with each waypoint in the path. This sequence contains the route information needed by the Trajectory Level for generation of the optimum trajectory, since it describes the actual track of the aircraft across the ground.

CONSTRUCTION OF DATA BASES

There are four data bases accessible by the Route Generator. The first data base is contained in the matrix called NAME, which holds the names of all the waypoints stored in alphabetical order. The position of each waypoint name within this array is very important, because the remaining data bases use this position as a pointer for identifying segments between waypoints within the connectivity matrix. The second data base, called the primary data base, contains all the information necessary to permit the Selector to choose the shortest route through the airway system from origin to destination. This includes the endpoints of each segment, the segment length, and the directions of travel permissible on each segment. The third data base contains the track information necessary for the Compiler to assemble the entire sequence of track angles and mileages for passage down to the Trajectory Level. The fourth data base contains the magnetic variation from true north associated with each waypoint.

Each entry in the primary data base lists general information on one particular airway segment. The entry is constructed entirely of integer numbers, and has the following format:

AAABBB CCC DDD E

where

AAA is a pointer to the location of the name of the first endpoint in the NAME array,

BBB is a pointer to the location of the name of the second endpoint in the NAME array,

CCC is the Jet Route designation number,

DDD is the length, in nautical miles, of that entire segment, and

E is a direction constraint flag indicating the allowable directions of travel on that segment:

0 - open (travel in both directions permissible)

1 - travel inbound to first endpoint only

2 - travel inbound to second endpoint only

For example, notice in figure 3 that Fort Lauderdale DME is named FLL. In this implementation FLL is located in the NAME array in position 21. Orlando VORTAC is named ORL, and is located in the NAME array in position 48. The segment connecting these two waypoints is designated J20, and has an overall length of 161 nautical miles. Therefore, this segment is stored in the primary data base as:

021048 020 161 0

The trailing zero is the direction constraint flag, indicating that this segment is normally open to travel in both directions.

Each entry in the third data base also lists information on only one airway segment, and is constructed of only integer numbers, but here it has the following format:

AAABBB FF GGGHHH IIIJJJ KKKLLL . . . etc.

where

AAA is a pointer to the location of the name of the first endpoint in the NAME array,

BBB is a pointer to the location of the name of the second endpoint in the NAME array,

FF is the number of different portions comprising the airway segment,

GGG is the track angle of the first portion,

HHH is the distance, in nautical miles, that the first track angle is to be followed

III is the track angle of the second portion, and

JJJ is the distance, in nautical miles, that the second track angle is to be followed . . . etc.

Also notice in figure 3 the path J20 takes between FLL and ORL. J20 first tracks 339 degrees for 92 nautical miles, and then tracks 334 degrees for 69 more nautical miles. This information for J20 is represented in the secondary data base as:

021048 02 339092 334069

Since the exact location of all waypoints and turning points must be known in order to calculate the track angles, the information for this data base must be gathered from other sources in addition to the Enroute High Altitude Charts which display the Jet Route System. The data base containing the magnetic variation from true north at each waypoint is stored in an array. Each element in the array has associated with it a VORTAC name and a character which is "W" to indicate a westerly magnetic variation and "E" for an easterly variation.

The designated airways of the Jet Route System are not the only available paths through the sky; however, they are the most desirable ones from the viewpoint of ATC. Special clearances are routinely granted for flights to cities not served by the System, and for flights between cities where use of the System would obviously present a hindrance to both the airline flight crew and the ATC staff. In commercial applications of this flight planning program, therefore, the data bases will need to be tailored to the existing structure of allowable routing for each individual airline.

RESULTS AND DISCUSSION

Several sample runs were made to demonstrate the capability of the program to produce the shortest route between waypoints. Various features of the route generator program are illustrated, including the capability of reversing direction and of specifying that certain flight segments are undesirable.

The first example illustrates a case which was run for the trip between Miami, Florida and Atlanta, Georgia. Figure 4 shows the interactive dialogue of the user with the route generator program and the output of the program. The user specifies the departure and arrival points and any undesirable segments. The program produces a list of the airways segments and corresponding mileages that make up the generated route. The waypoints listed include Miami VORTAC (MIA), LaBelle VORTAC (LBV), Lakeland VORTAC (LAL), and Atlanta VORTAC (ATL). The Jet Route segments listed include J43 between MIA and LBV, J73 between LBV and LAL, and J89 between LAL and ATL.

The second example is the reversal of the first example trip. The departure point is Atlanta and the destination is Miami. Figure 5 shows the results of this example, which shows the reversing function of the subroutine GNRATR. GNRATR adds 180 degrees to the track angles and reverse their order whenever travel occurs in the direction opposite from that in which the track and mileage information is stored.

Two sample runs that illustrate the ability of the program to plan around route segments that have been flagged as unacceptable are illustrated in figures 6 and 7. These runs demonstrate the fact that the program does always choose the shortest path between waypoints. Both cases were run for the trip between Hobby, Texas and Oklahoma City, Oklahoma. The waypoints listed include Hobby (HUB), Humble (IAH), Navasota (TNV), Dallas - Ft. Worth (DFW), and Oklahoma City (OKC). The Jet Route segments listed include J177 between HUB and IAH, J87 between IAH and TNV, J33 between IAH and DFW, J87 between TNV and DFW, and J21 between DFW and OKC.

CONCLUDING REMARKS

The computer program for generating a horizontal route that minimizes ground distance has been identified as part of the fault monitoring function of an onboard fault monitoring and diagnosis system. Using a small subset of the geographical reference points in the High Altitude Jet Route System, the program has been demonstrated to generate the shortest route between specified source and destination airports. The capability of the route generator program to route around path segments designated as undesirable was also demonstrated.

APPENDIX

Description, Recommendations, and Source Listing of Program ROUTEGEN and Subroutines LOADER, NOGOOD, SLECTR, GNRATR, and MAGVAR

The computer program described in this paper was written in FORTRAN 77 on a Prime 750 mini-computer, running under the Primos* operating system. A Tektronix 4014-1 was used for the remote terminal. The primary data base is stored on disk as the file named SEGDAT, and the secondary data base is stored as the file named TRACKDAT.

Program ROUTEGEN

Description

This is the driver program for the subroutines which perform the actual functions of selection and generation of the route. The origin and destination are read in from this level, and the selected route is output from this level.

Recommendations

1. The program currently communicates interactively with the user via a CRT terminal. If desired, points of departure and arrival could be pre-programmed or read in from another source, and the output could be printed out on a NAV display or stored for future reference by the pilot.

2. The call to the subroutine LOADER could be eliminated by storing all of the primary data base information in the appropriate arrays through the use of DATA statements. The execution time would definitely decrease, but the program size would increase. Depending on the overall size of the graph network, this may or may not be a desirable alternative.

Source Listing

The listing of program ROUTEGEN is as follows:

```
      PROGRAM ROUTEGEN
C
C--THIS PROGRAM IS THE DRIVER PROGRAM FOR THE ROUTINES WHICH WILL
C--CALCULATE THE SHORTEST ROUTE FROM POINT OF DEPARTURE TO POINT
C--OF ARRIVAL USING THE ESTABLISHED HIGH ALTITUDE JET ROUTE SYSTEM IN
C--EFFECT ON THE CONTINENTAL UNITED STATES, AND GENERATE THE TRACK
C--SEQUENCE NECESSARY TO COMPLETE THE PATH.
C
C--CURRENTLY, ONLY THE SOUTHEASTERN AND SOUTHCENTRAL UNITED STATES
C--PORTION OF THE JET ROUTE SYSTEM IS INCLUDED IN THE DATA BASE,
C--WHICH INCLUDES THE INITIALIZATION OF THE NAME ARRAY IN THE DATA
C--STATEMENT.
C
      COMMON /BLK1/JROUTE(70,70),LENGTH(70,70),IDIREC(70,70)
C
C     JROUTE - ARRAY CONTAINING THE JET ROUTE NAMES
C     LENGTH - ARRAY CONTAINING THE LENGTH (MILES) OF EACH SEGMENT
C     IDIREC - ARRAY OF FLAGS DESCRIBING DIRECTION CONSTAINTS
C                  0 - OPEN
C                  1 - INBOUND ONLY
C                  2 - OUTBOUND ONLY
C
C
      COMMON /BLK2/PATH(20),NSEGS,TOTLEN,SOURCE,SINK,N
C
C       PATH - ARRAY OF POINTERS TO VORTAC NAMES
C      NSEGS - NUMBER OF REQUIRED SEGMENTS TO DESTINATION
C     TOTLEN - TOTAL LENGTH (MILES) OF FINAL ROUTE
C     SOURCE - NODE CORRESPONDING TO POINT OF DEPARTURE
C       SINK - NODE CORRESPONDING TO POINT OF ARRIVAL
C          N - NUMBER OF NODES (VORTACS)
C
C
      COMMON /BLK3/IN,IOUT
C
C         IN - LOGICAL UNIT FOR INPUT FROM TERMINAL SCREEN
C       IOUT - LOGICAL UNIT FOR OUTPUT TO TERMINAL SCREEN
```

```
C
C
C       NAME - ARRAY CONTAINING NAMES OF ALL VORTACS (CHAR*3)
C      NAME1 - NAME OF SOURCE VORTAC
C      NAME2 - NAME OF SINK VORTAC
C
       CHARACTER*3 NAME(70),NAME1,NAME2,TEMP
       INTEGER PATH,TOTLEN,SOURCE,SINK
C
       DATA NAME/ 3HACT,  3HADM,  3HAEX,  3HAGS,  3HAMG,
      .           3HARG,  3HATL,  3HAUS,  3HBNA,  3HBRO,
      .           3HBSY,  3HCAE,  3HCEW,  3HCHA,  3HCHS,
      .           3HCRP,  3HCTY,  3HDAS,  3HDFW,  3HEYW,
      .           3HFLL,  3HFLO,  3HGRW,  3HGSO,  3HHUB,
      .           3HIAH,  3HIGB,  3HILM,  3HJAN,  3HJAX,
      .           3HLAL,  3HLBV,  3HLCH,  3HLEV,  3HLFK,
      .           3HLGC,  3HLIT,  3HMCB,  3HMCN,  3HMEI,
      .           3HMEM,  3HMGM,  3HMIA,  3HMQP,  3HMSY,
      .           3HOKC,  3HOMN,  3HORL,  3HPBI,  3HPIE,
      .           3HPSX,  3HRDU,  3HRMG,  3HRZC,  3HSAT,
      .           3HSAV,  3HSHV,  3HSJI,  3HSPA,  3HSPS,
      .           3HSRQ,  3HTAY,  3HTLH,  3HTNV,  3HTOC,
      .           3HTXK,  3HTYS,  3HVRB,  3HVUZ,  3HXXX/
       N=69
C
C--SET INPUT AND OUTPUT LOGICAL UNITS
       IN=1
       IOUT=1
C
C--DETERMINE INDEX OF THE SOURCE AND OF THE SINK
       WRITE(IOUT,10)
 10    FORMAT(/' ENTER DEPARTING AND ARRIVING CITIES')
       READ(IN,20)NAME1,NAME2
 20    FORMAT(A3,1X,A3)
       DO 30 I=1,N
          TEMP=NAME(I)
          IF(TEMP.EQ.NAME1)SOURCE=I
```

```
          IF(TEMP.EQ.NAME2)SINK=I
 30     CONTINUE
C
C--SUBROUTINE CALLS TO PERFORM ROUTE SELECTION
        CALL LOADER
        WRITE(IOUT,40)
 40     FORMAT(/' DO YOU WISH TO FLAG ANY UNDESIRABLE SEGMENTS?')
        READ(IN,50)IANS
 50     FORMAT(A1)
        IF(IANS.EQ.1HY)CALL NOGOOD(NAME)
        CALL SLECTR
C
C--PRINT OUT FINAL ROUTE TO SCREEN
        WRITE(IOUT,60)
 60     FORMAT(/' VORTAC   JET ROUTE')
        TOTLEN=0
        K=PATH(1)
        DO 80 I=2,NSEGS
           J=K
           K=PATH(I)
           WRITE(IOUT,70)NAME(J),JROUTE(J,K)
 70        FORMAT(2X,A3,7X,I3)
           TOTLEN=TOTLEN+LENGTH(J,K)
 80     CONTINUE
        WRITE(IOUT,90)NAME(K)
 90     FORMAT(2X,A3)
        WRITE(IOUT,100)TOTLEN
 100    FORMAT(/' TOTAL MILEAGE FOR PATH =',I5////)
C
C--SUBROUTINE CALL TO GENERATOR TO PRINT OUT TRACK DATA
        CALL GNRATR
C
        STOP
        END
```

APPENDIX

Subroutine LOADER

Description

This subroutine stores the information from the primary data base in the appropriate arrays. To minimize storage requirments, this information is given in the primary data base for travel in one direction only. That is, all the segments are stored heading generally south-to-north. As the information is being read into the different matrices, it is stored in both the south-to-north and the north-to-south positions.

Recommendations

1. The LENGTH array needs to be zeroed before the mileage information can be read in, and it has been done here using a DATA statement. This has the effect of increasing the storage requirements dramatically. A slower but more space efficient method would be to EQUIVALENCE the LENGTH array to a one-dimensional array, and then zero it using a single DO loop. If a machine-dependent function is available for this purpose it should be used in any implementation which still makes use of the LOADER subroutine.

Source Listing

The listing of subroutine LOADER is as follows:

```
      SUBROUTINE LOADER
C
C--THIS SUBROUTINE IS USED TO LOAD THE SPARSE LENGTH MATRIX USED
C--TO FIND THE SHORTEST ROUTE BETWEEN SOURCE AND SINK
C
      COMMON /BLK1/JROUTE(70,70),LENGTH(70,70),IDIREC(70,70)
      COMMON /BLK3/IN,IOUT
C
      DATA LENGTH/4900*0/
C
C--OPEN THE DATA FILE AND READ IN THE SPARSE MATRIX
      OPEN(31,FILE='SEGDAT',STATUS='OLD',ERR=899)
 10   READ(31,20,END=999)I,J,JROUTE(I,J),LENGTH(I,J),IDIREC(I,J)
 20   FORMAT(2I3,2(1X,I3),1X,I1)
C
C--SINCE THE MATRIX IS SYMMETRIC, DUPLICATE THE OTHER SIDE...
      JROUTE(J,I)=JROUTE(I,J)
      LENGTH(J,I)=LENGTH(I,J)
C
C--AND BE SURE TO SWITCH DIRECTION CONSTRAINT FLAG
      TEMP2=0
         TEMP1=IDIREC(I,J)
         IF(TEMP1.EQ.1)TEMP2=2
         IF(TEMP1.EQ.2)TEMP2=1
         IDIREC(J,I)=TEMP2
      GOTO 10
C
 899  WRITE(IOUT,*)' ERROR IN FILE OPENING.'
 999  CLOSE(31)
      RETURN
      END
```

APPENDIX

Subroutine NOGOOD

Description

This subroutine allows certain airway segments to be selected and labelled as unacceptable, for reasons of traffic, weather, or ATC constraints. The subroutine queries the user for the endpoints of each bad segment, and then inserts a zero for the mileage of that segment in the LENGTH matrix. This effectively removes the segment from consideration within the SLECTR subroutine.

Recommendations

1. All the information needed for labelling unacceptable segments could, if desired, be read in from a source other than interactive communication with a user. A realistic implementation of this program might have the traffic, weather, and ATC constraints stored in a dynamic data base, possibly being regularly updated by a data base monitor/modifier program. The Route Level Monitor would then need to periodically check the status of all segments in the planned route to determine if modification of the chosen flight plan is necessary.

Source Listing

The listing of subroutine NOGOOD is as follows:

```
      SUBROUTINE NOGOOD(NAME)
C
C--THIS SUBROUTINE IS USED TO FLAG UNUSABLE/UNDESIRABLE ROUTE SEGMENTS.
C
      COMMON /BLK1/JROUTE(70,70),LENGTH(70,70),IDIREC(70,70)
      COMMON /BLK2/PATH(20),NSEGS,TOTLEN,SOURCE,SINK,N
      COMMON /BLK3/IN,IOUT
C
      CHARACTER*3 NAME(70),NAME1,NAME2,TEMP
      INTEGER PATH,TOTLEN,SOURCE,SINK
C
      WRITE(IOUT,10)
 10   FORMAT(/' ENTER ENDPOINTS OF UNDESIRABLE SEGMENTS (XXX TO END)')
 20   READ(IN,30)NAME1,NAME2
 30   FORMAT(A3,1X,A3)
      IF(NAME1.EQ.3HXXX)GOTO 999
C
C--FIND THE INDEX OF EACH ENDPOINT
      DO 40 K=1,N
         TEMP=NAME(K)
         IF(TEMP.EQ.NAME1)I=K
         IF(TEMP.EQ.NAME2)J=K
 40   CONTINUE
C
C--LABEL THAT SEGMENT AS NO GOOD
      LENGTH(I,J)=0
      LENGTH(J,I)=0
      GOTO 20
C
 999  RETURN
      END
```

APPENDIX

Subroutine SLECTR

Description

This subroutine uses Dijkstra's algorithm (ref. 3) to find the path of least resistance (smallest total mileage) through the LENGTH array. Minor modification was necessary to incorporate the direction constraint flag capability, but the algorithm itself remains basically unaltered.

Source Listing

The listing of subroutine SLECTR is as follows:

```
      SUBROUTINE SLECTR
C
C--THIS SUBROUTINE PICKS THE SHORTEST PATH THROUGH THE LENGTH MATRIX,
C--AND STORES IT IN THE 'PATH' ARRAY.
C
      COMMON /BLK1/JROUTE(70,70),A(70,70),IDIREC(70,70)
      COMMON /BLK2/PATH(20),NSEGS,TOTLEN,SOURCE,SINK,N
C
      INTEGER PERM,TENT,STATE(70,3),A,PATH,TOTLEN,SOURCE,SINK,
     .        PREDECESSOR
      DATA PREDECESSOR/1/,LENGTH/2/,LABL/3/,INFINITY/9999/,
     .     PERM/1/,TENT/0/
C
C--INITIALIZE STATE MATRIX
      DO 10 I=1,N
         STATE(I,PREDECESSOR)=0
         STATE(I,LENGTH)=INFINITY
         STATE(I,LABL)=0
 10   CONTINUE
      STATE(SINK,LENGTH)=0
      STATE(SINK,LABL)=PERM
      K=SINK
C
C--IS THERE A BETTER PATH FROM K?
 20   CONTINUE
         DO 30 I=1,N
            IF((A(K,I).NE.0).AND.(STATE(I,LABL).EQ.TENT))THEN
               IF((STATE(K,LENGTH)+A(K,I).LT.STATE(I,LENGTH)).AND.
     .            (IDIREC(K,I).NE.2))THEN
                  STATE(I,PREDECESSOR)=K
                  STATE(I,LENGTH)=STATE(K,LENGTH)+A(K,I)
               ENDIF
            ENDIF
 30      CONTINUE
C
C--FIND THE TENTATIVELY LABELLED NODE WITH THE SMALLEST LABEL
         MIN=INFINITY
```

```
            K=0
            DO 40 I=1,N
               IF((STATE(I,LABL).EQ.TENT).AND.(STATE(I,LENGTH).LT.MIN))THEN
                  MIN=STATE(I,LENGTH)
                  K=I
               ENDIF
 40         CONTINUE
            STATE(K,LABL)=PERM
         IF(K.NE.SOURCE)GOTO 20      /* REPEAT UNTIL WE REACH THE SOURCE
C
C--COPY THE PATH INTO THE OUTPUT ARRAY
         K=SOURCE
         I=0
 50      CONTINUE
            I=I+1
            PATH(I)=K
            K=STATE(K,PREDECESSOR)
         IF(K.NE.0)GOTO 50
         NSEGS=I
C
         RETURN
         END
```

APPENDIX

Subroutine GNRATR

Description

This subroutine compiles the track angle and mileage sequence from the third data base. The current technique used for this purpose employs a simple, but slow, process of searching the entire data base for each list of segment information. As each segment list is located, it is printed out to the CRT screen according to the direction of travel. If travel occurs in the direction opposite that in which the segment information is stored, 180 degrees are added to the track angles, which are output in reverse order with their mileage.

Recommendations

1. A much faster search method would include storing the airway segment data in the same sequence within both data bases. An array of pointers to the location of each chosen segment could be filled at the same time as the array of pointers to the waypoint names, known as the PATH array. The track and mileage information could be compiled much faster with the few number of disk file operations required. Once assembled, the subroutine GNRATR currently merely prints the track and mileage information to the CRT terminal screen and to a file for later use by the Trajectory Level. When combined with other computer programs that perform the functions of the other levels in the fault monitoring and diagnosis segment's basic architecture, this information will be used by the Trajectory Level.

Source Listing

The listing of subroutine GNRATR is as follows:

```
      SUBROUTINE GNRATR
C
C--THIS SUBROUTINE RETRIEVES FROM THE DATA BASE THE ENTIRE SEQUENCE
C--OF TRACK ANGLES (NO WIND TRUE HEADINGS) AND CORRESPONDING
C--DISTANCES THAT IS NEEDED TO TRAVEL ALONG THE PATH PREVIOUSLY
C--CHOSEN BY THE SELECTOR.
C
      COMMON /BLK2/PATH(20),NSEGS,TOTLEN,SOURCE,SINK,N
      COMMON /BLK3/IN,IOUT
      DIMENSION TRACK(15),DIST(15)
C
C     TRACK - ARRAY OF TEMPORARY TRACK ANGLES
C      DIST - ARRAY OF TEMPORARY CORRESPONDING MILEAGES
C
C
      INTEGER PATH,TOTLEN,SOURCE,SINK,TRACK,DIST,TEMP1,TEMP2,TEMP3
C
C--OPEN THE DATA FILE, AND LOOP TO LOCATE DESIRED SEGMENTS
      OPEN(31,FILE='TRACKDAT',STATUS='OLD',ERR=899)
      JNDEX=PATH(1)
      DO 60 I=2,NSEGS
         INDEX=JNDEX
         JNDEX=PATH(I)
         REWIND(31)
 10      READ(31,20,END=60)K,L,NTRACKS,(TRACK(M),DIST(M),M=1,NTRACKS)
 20      FORMAT(2I3,1X,I2,15(1X,2I3))
C
C--IF THIS IS THE NEXT SEGMENT, PRINT OUT THE TRACK INFO
         IF((K.EQ.INDEX).AND.(L.EQ.JNDEX))THEN
            DO 40 J=1,NTRACKS
               WRITE(IOUT,30)TRACK(J),DIST(J)
 30            FORMAT(' TRACK =',I4,10X,'DISTANCE =',I4)
 40         CONTINUE
         ELSE IF((L.EQ.INDEX).AND.(K.EQ.JNDEX))THEN
            DO 50 J=1,NTRACKS
               TEMP1=TRACK(J)
                  TEMP2=TEMP1+180
```

```
                  IF(TEMP2.GT.360)TEMP2=TEMP2-360
                TEMP3=NTRACKS-J+1
                WRITE(IOUT,30)TEMP2,DIST(TEMP3)
 50          CONTINUE
          ENDIF
C
C--IF NOT, THEN READ THE NEXT LINE IN THE DATA BASE
          GOTO 10
C
 60    CONTINUE
       GOTO 999
C
 899   WRITE(IOUT,*)' ERROR IN FILE OPENING.'
 999   CLOSE(31)
       RETURN
       END
```

APPENDIX

Subroutine MAGVAR

Description

This subroutine retrieves the magnetic variation of each point along the path from true north. The data base is searched using a simple sequential search until the desired waypoint names are found. The magnetic variation for each waypoint in the chosen path is stored in an array called VAR_MAG.

Source Listing

The listing of MAGVAR is as follows:

```
      SUBROUTINE MAGVAR(VORTAC,DIRECTION,VAR_MAG,VOR_NUMBER)
C=====================================================
C THIS SUBROUTINE READS THE MAGNETIC VARIATION FROM A
C FILE OF VARIATIONS. TWO ARRAYS NAMED DIRECTION(LATITUDE)
C AND VAR_MAG ARE RETURNED.
C=====================================================
      CHARACTER DIRECTION(10)
      CHARACTER*3 VORTAC(10)
      CHARACTER*3 ALPHA
      REAL VAR_MAG(10)
      REAL MAG
      CHARACTER A
      INTEGER VOR_NUMBER
C
C
C
      COUNT = 0
      OPEN(30,FILE='VARDAT',STATUS='OLD')
      REWIND(30)
      DO 40 I = 1,100
        READ(30,20,END=100) ALPHA,MAG,A
20      FORMAT(A3,1X,F3.1,1X,A1)
        DO 30 J = 1,VOR_NUMBER
          IF(ALPHA.EQ.VORTAC(J)) THEN
             VAR_MAG(J) = MAG
             DIRECTION(J) = A
             COUNT = COUNT + 1
          ENDIF
30      CONTINUE
      IF(COUNT.EQ.VOR_NUMBER) GOTO 99
40    CONTINUE
99    CLOSE(30)
      RETURN
100   WRITE(6,101)
101   FORMAT(1X,'+++ EOF FOUND IN VARIATION DATA FILE +++')
      RETURN
      END
```

REFERENCES

1. Chien, R. T.; Chen, D. C.; Ho, W. P. C.; and Pan, Y. C.: Multilevel Semantic Analysis and Problem-Solving in the Flight Domain. NASA Grant NCC1-52, University of Illinois at Urbana-Champaign, NASA CR-169282, August 1982.

2. Winston, Patrick H. and Horn, Berthold, K. P.: Lisp. Addison-Wesly Publishing Co., 1981.

3. Tanenbaum, Andrew S.: Computer Networks. Prentice Hall, Inc., 1981, pp. 36-41.

4. United States Government Flight Information Publication, Enroute High Altitude. U. S. National Oceanic and Atmospheric Administration, U. S. Department of Commerce, October 28, 1982 - December 23, 1982.

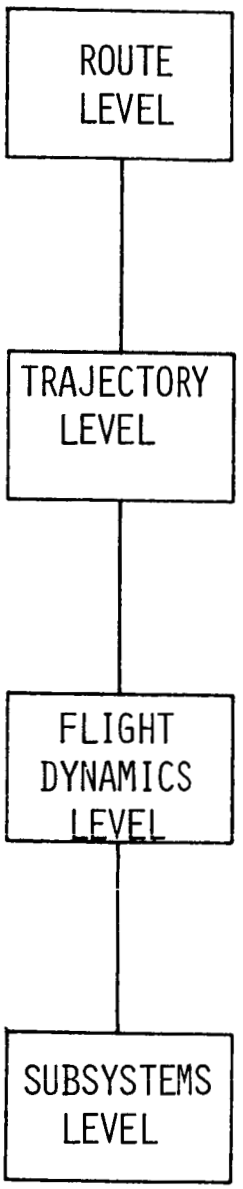

Figure 1.- Multi-level architecture of the fault monitor.

Waypoints	IAH	IGB	ILM	JAN	JAX	LAL	LBU	LCH	LEU	LFK	LGC	LIT	MCB	MCN	MEI
IAH	0	0	0	0	0	0	0	117	278	79	0	0	0	0	0
IGB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ILM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JAN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	70
JAX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LAL	0	0	0	0	0	0	77	0	0	0	0	0	0	0	0
LBU	0	0	0	0	0	77	0	0	0	0	0	0	0	0	0
LCH	117	0	0	0	0	0	0	0	0	0	0	0	163	0	0
LEU	278	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LFK	79	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LGC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LIT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MCB	0	0	0	0	0	0	0	163	0	0	0	0	0	0	98
MCN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MEI	0	0	0	70	0	0	0	0	0	0	0	0	98	0	0

Numbers indicate nautical miles between waypoints.

Figure 2.- Partial printout of the contents of the graph matrix, illustrating the connectivity concept between waypoints.

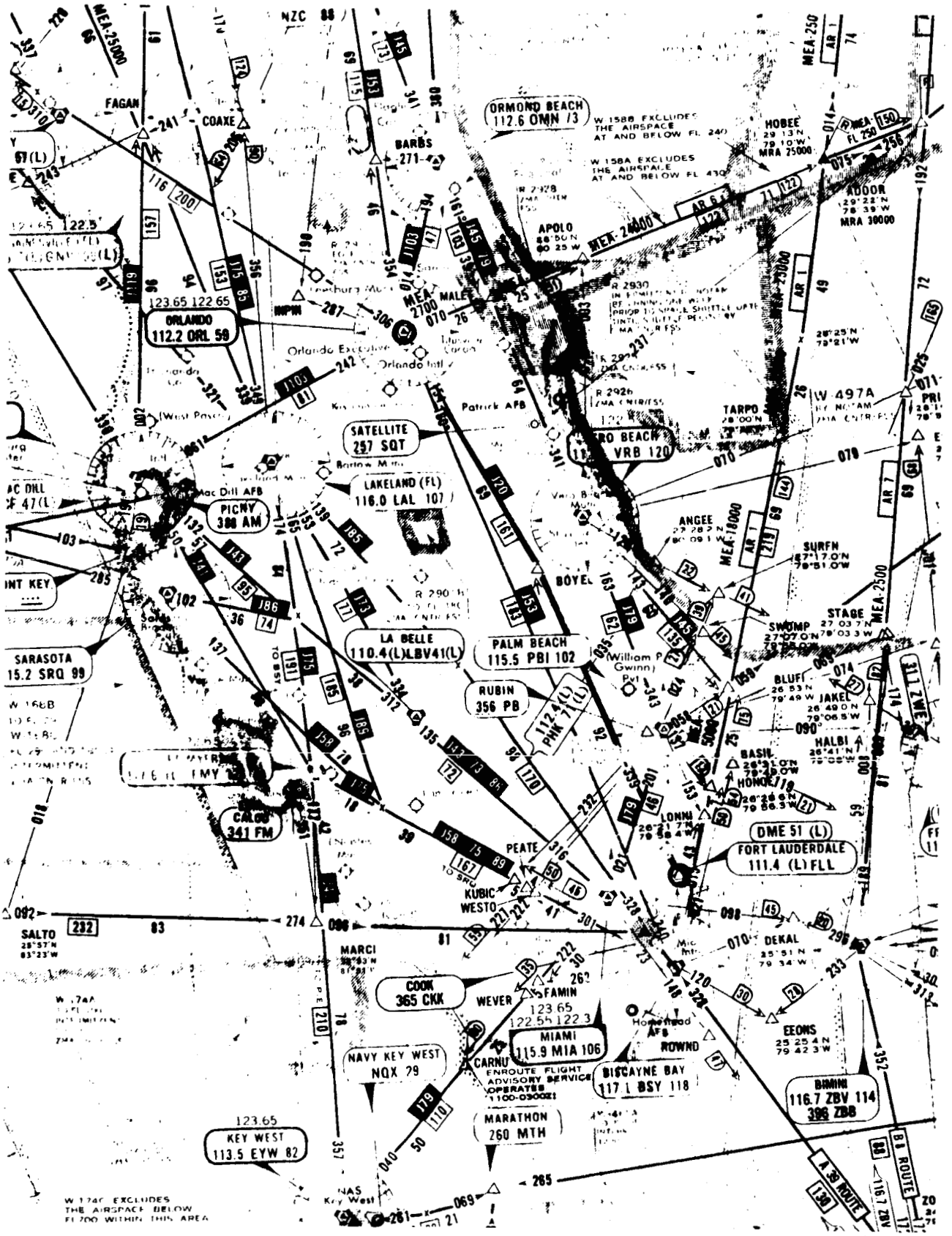

Figure 3.- South Florida portion of Enroute High Altitude.

```
 ENTER DEPARTING AND ARRIVING CITIES
MIA ATL

 DO YOU WISH TO FLAG ANY UNDESIRABLE SEGMENTS?
N

 VORTAC    JET ROUTE
  MIA          43
  LBV          73
  LAL          89
  ATL

 TOTAL MILEAGE FOR PATH =  509



 TRACK = 316          DISTANCE =  72
 TRACK = 334          DISTANCE =  77
 TRACK = 339          DISTANCE =  94
 TRACK = 339          DISTANCE =  66
 TRACK = 339          DISTANCE =  20
 TRACK = 339          DISTANCE =  30
 TRACK = 339          DISTANCE = 150
**** STOP
```

Figure 4.- Example dialogue and output for the trip from Miami, Florida to Atlanta, Georgia.

```
 ENTER DEPARTING AND ARRIVING CITIES
ATL MIA

 DO YOU WISH TO FLAG ANY UNDESIRABLE SEGMENTS?
N

 VORTAC    JET ROUTE
  ATL         89
  LAL         73
  LBV         43
  MIA

 TOTAL MILEAGE FOR PATH =  509



 TRACK = 159          DISTANCE = 150
 TRACK = 159          DISTANCE =  30
 TRACK = 159          DISTANCE =  20
 TRACK = 159          DISTANCE =  66
 TRACK = 159          DISTANCE =  94
 TRACK = 154          DISTANCE =  77
 TRACK = 136          DISTANCE =  72
**** STOP
```

Figure 5.- Example dialogue and output for the trip from Atlanta, Georgia to Miami, Florida.

```
 ENTER DEPARTING AND ARRIVING CITIES
HUB OKC

 DO YOU WISH TO FLAG ANY UNDESIRABLE SEGMENTS?
N

 VORTAC    JET ROUTE
  HUB         177
  IAH          33
  DFW          21
  OKC

 TOTAL MILEAGE FOR PATH =  381



 TRACK = 343          DISTANCE =  19
 TRACK = 341          DISTANCE =  54
 TRACK = 341          DISTANCE =  47
 TRACK = 310          DISTANCE = 100
 TRACK = 347          DISTANCE =  81
 TRACK = 329          DISTANCE =  80
**** STOP
```

Figure 6.- Example dialogue and output for the trip from Hobby, Texas to Oklahoma City, Oklahoma.

```
 ENTER DEPARTING AND ARRIVING CITIES
HUB OKC

 DO YOU WISH TO FLAG ANY UNDESIRABLE SEGMENTS?
Y

 ENTER ENDPOINTS OF UNDESIRABLE SEGMENTS (XXX TO END)
IAH DFW
XXX

 VORTAC    JET ROUTE
  HUB         177
  IAH          87
  TNV          87
  DFW          21
  OKC

 TOTAL MILEAGE FOR PATH =  384



 TRACK = 343          DISTANCE =  19
 TRACK = 290          DISTANCE =  42
 TRACK = 334          DISTANCE =  15
 TRACK = 334          DISTANCE =  40
 TRACK = 335          DISTANCE =  22
 TRACK = 334          DISTANCE =  85
 TRACK = 347          DISTANCE =  81
 TRACK = 329          DISTANCE =  90
**** STOP
```

Figure 7.- Example dialogue and output for the trip from Hobby, Texas to Oklahoma City Oklahoma with undesirable segments specified.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-86264		

4. Title and Subtitle	5. Report Date
A Route Generator Concept for Aircraft Onboard Fault Monitoring	August 1984
	6. Performing Organization Code: 505-35-13-04

7. Author(s)	8. Performing Organization Report No.
Michael T. Palmer Kathy H. Abbott	
	10. Work Unit No.
9. Performing Organization Name and Address	
NASA Langley Research Center Hampton, VA 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered: Technical Memorandum
12. Sponsoring Agency Name and Address	
National Aeronautics and Space Administration Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Because of the increasingly complex environments in which the flight crews of commercial aviation aircraft must operate, a research effort is currently underway at NASA Langley Research Center to investigate the potential benefits of intelligent cockpit aids, and to establish guidelines for the application of artificial intelligence techniques to advanced flight management concepts. The segment of this research area that concentrates on automated fault monitoring and diagnosis requires that a reference frame exist, against which the current state of the aircraft may be compared to determine the existence of a fault. This paper describes a computer program which generates the position of that reference frame that specifies the horizontal flight route.

17. Key Words (Suggested by Author(s))
flight management
artificial intelligence
route planning

18. Distribution Statement
Unclassified - Unlimited
Subject Category 04

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	32	A03

N-305

For sale by the National Technical Information Service, Springfield, Virginia 22161